Programming Rubric v1.1

Programmer:_____Problem:_____Date Due:_____

Criteria		Exceptionally well executed (10)	Good, with room for improvement (8)	Meets minimum requirement (6)	Pts (you)	Pts (me)
Documentation	**Assignment:**	Assignment turned in on time and neatly with all sections clearly labeled and stapled together in the correct order.	Assignment up to 24 hours late but otherwise turned in correctly.	Assignment up to 72 hours late or turned in incorrectly.		
	Specification	Problem is clearly defined. Specification is complete and appropriately detailed. Given values and user inputs are explained.	Problem is defined. Specification is mostly complete, but perhaps not entirely appropriately detailed. Given values and user inputs are listed.	Problem definition is deficient in some way, or specification does not adequately represent the problem.		
	Top-Down Design	Top-down design method followed and written in appropriate detail.	Top-down method followed, but level of detail is too vague or too exact.	Top-down design method attempted, but poorly executed.		
	Test Cases	Clear and well thought out test cases presented that cover all boundary conditions and a comprehensive range of user inputs.	Good test cases, but some boundary conditions are missing. Range of user input mostly well thought out.	Little in the way of test cases. One or more obvious boundary conditions missing.		
Source Code	**Modularization & Generalization**	Program broken into well thought out elements that are of an appropriate length, scope and independence. Individual elements are written in a way that actively invites reuse in other projects.	Code elements are generally well planned and executed. Some code is repeated that should be encapsulated. Individual elements are often, but not always, written in a way that invites code reuse.	Code elements exist, but are not well thought out, are used in a somewhat arbitrary fashion, or do not improve program clarity. Elements are seldom written in a way that invites code reuse.		
	Design, Structure & Efficiency	Program is designed in a clear and logical manner. Control structures are used correctly. The most appropriate algorithms are implemented.	Program is mostly clear and logical. Control structures are used correctly. Reasonable algorithms are implemented.	Program isn't as clear or logical as it should be. Control structures are occasionally used incorrectly. Steps that are clearly inefficient are used.		
	Readability, Consistency & Naming	Coding style guidelines are followed correctly, code is exceptionally easy to read and maintain. All names are consistent with regard to style and are expressive without being verbose.	Coding style guidelines are almost always followed correctly. Code is easy to read. Names are consistent in style and expressive. Isolated cases may be verbose, overly terse or ambiguous.	Coding style guidelines are not followed and/or code is less readable than it should be. Names are nearly always consistent, but occasionally verbose, overly terse, ambiguous or misleading.		
	Initial Comments	Initial comments are complete. Internal documentation is complete and well suited to the program	Initial comments are complete but internal documentation is in some small fashion inadequate.	Initial comments are incomplete or internal documentation is inadequate.		
	Coding Comments	Comments clarify meaning where needed.	Comments usually clarity meaning. Unhelpful comments may exist.	Comments exist, but are frequently unhelpful or occasionally misleading.		
Execution	**User Interface**	Screen based instructions and final output are clear, correct and attractive. Program is "user friendly" with informative and consistent prompts and messages.	Screen based instructions and final output are mostly clear, correct and attractive. Program is "user friendly" with informative and consistent prompts and messages.	Screen based instructions and final output are not clear, are not correct or are not attractive. And/or Program is not "user friendly.		
	Robustness	Program handles erroneous or unexpected input gracefully; action is taken without surprising the user.	All obvious error conditions are checked for and appropriate action is taken.	Some obvious error conditions are checked for and some sort of action is taken.		
	Testing & Correctness	Testing is complete without being redundant. All boundary cases are considered, tested and work correctly.	All key items are tested, but testing may be redundant. Nearly all boundary cases are considered, tested and work correctly.	Testing was done, but is not sufficiently complete. Most boundary cases are considered, tested and work correctly.		
	Degree of Difficulty (13 points)	Degree of Difficulty optional challenges can be found at the bottom of program specification sheets. They are optional in that you can successfully turn in programs without attempting any of them, however completing them represents the difference between A level work and B level work. Also, doing them generally makes the instructor think that you're a really hoopy frood.				
Summarize all degree of difficulty bonuses (write "see back" if they are listed on the back)				**Total Points (out of 133)**		

Make sure you look at your specification, pseudo-code, test cases, source-code and testing for additional comments.